```
EEEEEEEEEEEEEEE  RRRRRRRRRRRR     RRRRRRRRRRRR     FFFFFFFFFFFFFFF  MMM         MMM  TTTTTTTTTTTTTTT
EEEEEEEEEEEEEEE  RRRRRRRRRRRR     RRRRRRRRRRRR     FFFFFFFFFFFFFFF  MMM         MMM  TTTTTTTTTTTTTTT
EEEEEEEEEEEEEEE  RRRRRRRRRRRR     RRRRRRRRRRRR     FFFFFFFFFFFFFFF  MMM         MMM  TTTTTTTTTTTTTTT
EEE              RRR         RRR  RRR         RRR  FFF              MMMMMMM   MMMMMMM        TTT
EEE              RRR         RRR  RRR         RRR  FFF              MMMMMMM   MMMMMMM        TTT
EEE              RRR         RRR  RRR         RRR  FFF              MMMMMMM   MMMMMMM        TTT
EEE              RRR         RRR  RRR         RRR  FFF              MMM MMM MMM  MMM        TTT
EEE              RRR         RRR  RRR         RRR  FFF              MMM MMM MMM  MMM        TTT
EEE              RRR         RRR  RRR         RRR  FFF              MMM MMM MMM  MMM        TTT
EEEEEEEEEEEE     RRRRRRRRRRRR     RRRRRRRRRRRR     FFFFFFFFFFF      MMM     MMM  MMM        TTT
EEEEEEEEEEEE     RRRRRRRRRRRR     RRRRRRRRRRRR     FFFFFFFFFFF      MMM     MMM  MMM        TTT
EEEEEEEEEEEE     RRRRRRRRRRRR     RRRRRRRRRRRR     FFFFFFFFFFF      MMM     MMM  MMM        TTT
EEE              RRR   RRR         RRR   RRR        FFF              MMM     MMM  MMM        TTT
EEE              RRR   RRR         RRR   RRR        FFF              MMM     MMM  MMM        TTT
EEE              RRR   RRR         RRR   RRR        FFF              MMM     MMM  MMM        TTT
EEE              RRR       RRR     RRR       RRR    FFF              MMM     MMM  MMM        TTT
EEE              RRR       RRR     RRR       RRR    FFF              MMM     MMM  MMM        TTT
EEE              RRR       RRR     RRR       RRR    FFF              MMM     MMM  MMM        TTT
EEEEEEEEEEEEEEE  RRR           RRR  RRR           RRR  FFF              MMM     MMM  MMM        TTT
EEEEEEEEEEEEEEE  RRR           RRR  RRR           RRR  FFF              MMM     MMM  MMM        TTT
EEEEEEEEEEEEEEE  RRR           RRR  RRR           RRR  FFF              MMM     MMM  MMM        TTT
```

```
EEEEEEEEEE  RRRRRRRR    RRRRRRRR    FFFFFFFFFF  MM      MM  TTTTTTTTTT
EEEEEEEEEE  RRRRRRRR    RRRRRRRR    FFFFFFFFFF  MM      MM  TTTTTTTTTT
EE          RR      RR  RR      RR  FF          MMMM  MMMM      TT
EE          RR      RR  RR      RR  FF          MMMM  MMMM      TT
EE          RR      RR  RR      RR  FF          MM  MM  MM      TT
EE          RR      RR  RR      RR  FF          MM  MM  MM      TT
EEEEEEEE    RRRRRRRR    RRRRRRRR    FFFFFFF     MM      MM      TT
EEEEEEEE    RRRRRRRR    RRRRRRRR    FFFFFFF     MM      MM      TT
EE          RR  RR      RR  RR      FF          MM      MM      TT
EE          RR  RR      RR  RR      FF          MM      MM      TT
EE          RR    RR    RR    RR    FF          MM      MM      TT          ....
EE          RR    RR    RR    RR    FF          MM      MM      TT          ....
EEEEEEEEEE  RR      RR  RR      RR  FF          MM      MM      TT          ....
EEEEEEEEEE  RR      RR  RR      RR  FF          MM      MM      TT          ....

LL              IIIIII      SSSSSSSS
LL              IIIIII      SSSSSSSS
LL                II      SS
LL                II      SS
LL                II      SS
LL                II      SS
LL                II        SSSSSS
LL                II        SSSSSS
LL                II              SS
LL                II              SS
LL                II              SS
LL                II              SS
LLLLLLLLLL      IIIIII    SSSSSSSS
LLLLLLLLLL      IIIIII    SSSSSSSS
```

Table of contents

```
0000     1          .TITLE  ERRFMT
0000     2          .IDENT  'V04-000'
0000     3
0000     4  ;
0000     5  ;**********************************************************************************
0000     6  ;*                                                                                *
0000     7  ;*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                       *
0000     8  ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                        *
0000     9  ;*  ALL RIGHTS RESERVED.                                                          *
0000    10  ;*                                                                                *
0000    11  ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED          *
0000    12  ;*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE        *
0000    13  ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER        *
0000    14  ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY        *
0000    15  ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY        *
0000    16  ;*  TRANSFERRED.                                                                  *
0000    17  ;*                                                                                *
0000    18  ;*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE         *
0000    19  ;*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT        *
0000    20  ;*  CORPORATION.                                                                  *
0000    21  ;*                                                                                *
0000    22  ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS         *
0000    23  ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                       *
0000    24  ;*                                                                                *
0000    25  ;*                                                                                *
0000    26  ;**********************************************************************************
0000    27  ;
0000    28
0000    29  ;++
0000    30  ; FACILITY:  ERROR LOG FORMAT PROGRAM
0000    31  ;
0000    32  ; ABSTRACT:  THIS PROGRAM EMPTIES THE ERROR LOG BUFFERS AND CREATES
0000    33  ;              A FILE, ERRLOG.SYS, IN A FORMAT ACCEPTABLE TO ERF.
0000    34  ;
0000    35  ;
0000    36  ; ENVIRONMENT:
0000    37  ;
0000    38  ; AUTHOR:  KATHLEEN D. MORSE,          CREATION DATE:  29-JUN-1977
0000    39  ;
0000    40  ; MODIFIED BY:
0000    41  ;
0000    42  ;       V03-011 TCM0008         Trudy C. Matthews       20-Aug-1984
0000    43  ;               Increase the size of the buffer that we use to receive
0000    44  ;               messages from the venus console during creation of the
0000    45  ;               cpu-specific error log.  Change filename of error snapshot
0000    46  ;               file on the console device from SNAPx.LOG to SNAPx.DAT.
0000    47  ;
0000    48  ;       V03-010 TCM0007         Trudy C. Matthews       19-Jul-1984
0000    49  ;               Add ability to create a CPU-specific errorlog once per
0000    50  ;               initialization of the ERRFMT process.
0000    51  ;
0000    52  ;       V03-009 EAD0171         Elliott A. Drayton      7-May-1984
0000    53  ;               Replace $UPDATE to support time stamps.
0000    54  ;
0000    55  ;       V03-008 EAD0139         Elliott A. Drayton      11-Apr-1984
0000    56  ;               Changed output FAB to allow shared read access.
0000    57  ;
```

```
0000     58 ;        V03-007 CWH1002         CW Hobbs                1-Mar-1983
0000     59 ;                Convert the errlog pid to an extended pid for the $delprc.
0000     60 ;
0000     61 ;        V03-006 TCM0006         Trudy C. Matthews       30-Dec-1982
0000     62 ;                Fix bug in ERF$TIMSTMP; it modifies R2 but doesn't save the
0000     63 ;                old value.
0000     64 ;
0000     65 ;        V03-005 TCM0005         Trudy C. Matthews       16-Jul-1982
0000     66 ;                Fix problem in V03-004 that caused a new ERRLOG.SYS to be
0000     67 ;                created each time the system was re-booted.
0000     68 ;
0000     69 ;        V03-004 TCM0004         Trudy C. Matthews       24-Jun-1982
0000     70 ;                When opening ERRLOG.SYS, check that its the same file we
0000     71 ;                accessed the last time.  If not, create a new version.
0000     72 ;
0000     73 ;        V03-003 ROW0080         Ralph O. Weber          08-APR-1982
0000     74 ;                Move DEVFAO control string so that it is not in the middle
0000     75 ;                of the "ERROR ACCESSING ERROR LOG FILE" message text.
0000     76 ;
0000     77 ;        V03-002 STJ0251         Steven T. Jeffreys      01-Apr-1982
0000     78 ;                Do not send mount/dismount notification messages to
0000     79 ;                OPCOM depending on the appropriate sysgen parameter.
0000     80 ;
0000     81 ;        V03-001 STJ0228         Steven T. Jeffreys      19-Mar-1982
0000     82 ;                Use full device name when calling $GETDVI.
0000     83 ;
0000     84 ;        V02-012 LMP0014         L. Mark Pilant,         16-Mar-1982  11:15
0000     85 ;                Fix a problem with the setting of the desired operator
0000     86 ;                bits.  Also, fix a problem with using EFN 0 with GETDVI.
0000     87 ;
0000     88 ;        V02-011 LMP0007         L. Mark Pilant          13-Jan-1982   9:55
0000     89 ;                Notify the appropriate operators when volume mount and
0000     90 ;                dismount messages area seen.
0000     91 ;
0000     92 ;        V02-010 SPF0045         Steve Forgey            28-Dec-1981
0000     93 ;                Synchronize buffer copy with allocation interlock flag.
0000     94 ;
0000     95 ;        V02-009 PHL0013         Peter H. Lipman         21-Aug-1981
0000     96 ;                Change the output file specification for the error log file
0000     97 ;                to use the new system wide logical name SYS$ERRORLOG whic
0000     98 ;                is the [SYSERR] directory on the system disk.
0000     99 ;
0000    100 ;        V02-008 TCM0003         Trudy C. Matthews       6-Aug-1981
0000    101 ;                Change message sent to oerator's terminal when ERRFMT
0000    102 ;                deletes itself.
0000    103 ;
0000    104 ;        V02-007 KDM0059         Kathleen D. Morse       22-Jul-1981
0000    105 ;                Fix new file error log message.
0000    106 ;
0000    107 ;        V02-006 KDM0057         Kathleen D. Morse       15-Jul-1981
0000    108 ;                Add SID to error log buffer message format and make the
0000    109 ;                header fields be negative offsets from the message text.
0000    110 ;
0000    111 ;        V02-005 TCM0002         Trudy C. Matthews       13-Jul-1981
0000    112 ;                Document use of @SYS$SYSTEM:STARTUP ERRFMT instead of
0000    113 ;                @SYS$MANAGER:ERFSTART to re-start ERRFMT process after it
0000    114 ;                deletes itself.
```

```
0000    115 ;
0000    116 ;
0000    117 ;       V02-004 STJ0024         Steven T. Jeffreys      01-Feb-1981
0000    118 ;               Fixed bugs in mailbox write logic.
0000    119 ;
0000    120 ;
0000    121 ;--
```

```
                        0000    123          .SBTTL  DECLARATIONS
                        0000    124 ;
                        0000    125 ; INCLUDE FILES:
                        0000    126 ;
                        0000    127
                        0000    128 ;
                        0000    129 ; MACROS:
                        0000    130 ;
                        0000    131 ;
                        0000    132 ; EQUATED SYMBOLS:
                        0000    133 ;
                        0000    134          $PRDEF                          ; DEFINE PROCESSOR REGISTERS
                        0000    135          $DCDEF                          ; DEFICE DEVICE CLASS TYPES
                        0000    136          $DIBDEF                         ; DEVICE INFORMATION BUFFER
                        0000    137          $DVIDEF                         ; $GETDVI MESSAGE CODES
                        0000    138          $EMBETDEF                       ; ERROR MESSAGE ENTRY TYPES
                        0000    139          $EMBDEF                         ; DEFINE ERROR MESSAGE BUFFER HEADER
                        0000    140          $EMBTSDEF                       ; DEFINE TIME STAMP DEFINTIONS
                        0000    141          $ERFHDDEF                       ; ERROR FORMAT HEADER DEFINITIONS
                        0000    142          $ERFTSDEF                       ; ERROR FORMAT TIME STAMP DEFINTIONS
                        0000    143          $ERFVMDEF                       ; ERROR FORMAT VOLUME MOUNT DEFINITIONS
                        0000    144          $ERLDEF                         ; SYSTEM ERROR LOGING DEFINTIONS
                        0000    145          $OPCDEF                         ; OPERATOR MESSAGE DEFINITIONS
                        0000    146          $PCBDEF                         ; PROCESS CONTROL BLOCK DEFINITIONS
                        0000    147          $SSDEF                          ; DEFINE STATUS CODES
                        0000    148
00000002                0000    149 ERM$C_FORMAT = 2                         ; FORMAT NUMBER FOR VAX
000000FF                0000    150 ERF$C_LOOP_CNT = 255                     ; TIMES TO WAIT FOR BUFFER
00000258                0000    151 ERF$K_DLTA_STMP = <60*10>                ; TIME STAMP DELTA IN SECS
FF676980                0000    152 ERF$K_CLK_TICK = -<10*1000*1000>         ; CONVERSION TO CLOCK TICKS/SEC
                        0000    153
                        0000    154 ;
                        0000    155 ; OWN STORAGE:
                        0000    156 ;
                        0000    157
                  00000000      158          .PSECT  DATA,RD,WRT,NOEXE,PAGE
                        0000    159
00000200                0000    160 INBUF:   .BLKB   512                     ; INPUT BUFFER
                        0200    161 OUTFAB: $FAB     -                       ; RECORD ACCESS BLOCK
                        0200    162                  FAC=<PUT,UPD>,-         ; PUT AND UPDATE FILE ACCESS
                        0200    163                  FNA=OUTNAM,-            ; FILE NAME ADDRESS
                        0200    164                  FNS=OUTNAMSZ,-          ; LENGTH OF FILE NAME
                        0200    165                  NAM=NAMEBLOCK, -        ; ASSSOCIATED NAME BLOCK
                        0200    166                  RFM=VAR, -
                        0200    167                  FOP=CIF, -
                        0200    168                  SHR=<GET,UPI>, -
                        0200    169                  ORG=SEQ,-               ; SEQUENTIAL ORGANIZATION
                        0200    170                  MRS=0                   ; MAX RECORD SIZE UNSPECIFIED
                        0250    171
                        0250    172 OUTRAB: $RAB     -                       ; RECORD ACCESS BLOCK
                        0250    173                  ROP=<EOF,WBH>, -        ; OPEN TO END OF FILE
                        0250    174                  MBC=1, -
                        0250    175                  MBF=2, -
                        0250    176                  RAC=SEQ, -
                        0250    177                  FAB=OUTFAB              ; FILE ACCESS BLOCK ADDR
                        0294    178
                        0294    179 NAMEBLOCK:                               ; NAME BLOCK ASSOCIATED WITH OUTFAB
```

```
                                0294   180          $NAM
                    0000'0000'0000' 02F4   181 OUTFID:          .WORD   0[3]                    ; SAVED FILE ID
                                00  02FA   182 LASTENTRY:       .BYTE 0                         ; ENTRY TYPE OF LAST RECORD WRITTEN
                          00000000  02FB   183 SID:             .LONG   0                       ; SYSTEM ID #
                              0000  02FF   184 ERF$W_MBXCHN:    .WORD   0                       ; DIAGNOSTIC MAILBOX CHANNEL
                              0000  0301   185 ERF$W_MBXSIZ:    .WORD   0                       ; DIAGNOSTIC MAILBOX SIZE
                              0000  0303   186 ERF$W_MBXUNT:    .WORD   0                       ; PREVIOUS DIAG MBX UNIT #
                                    0305   187
55 21 43 41 21 5F 0000030D'010E0000' 0305   188 DEVFAO: .ASCID  /_!AC!UW:/                      ; $FAO control string to format device
                                3A 57  0313
                                    0315   189
                                    0315   190 ;
                                    0315   191 ; MESSAGE SENT TO OPERATOR UPON FAILURE TO WRITE TO ERROR LOG FILE.
                                    0315   192 ;
                                    0315   193 OPRMSG_DSC:
                                    0315   194 OPRMSG_LEN:
                    00000031'       0315   195          .LONG   OPRMSG_END-OPRMSG              ; SIZE OF OPERATOR MESSAGE BUFFER
                    00000325'       0319   196          .LONG   OPRMSG                         ; ADDRESS OF OPERATOR MESSAGE BUFFER
                                    031D   197 ROMSG_DSC:
                    00000100'       031D   198          .LONG   ROMSG_END-ROMSG
                    00000356'       0321   199          .LONG   ROMSG
                                    0325   200 OPRMSG:
                    00000103        0325   201          .LONG   OPC$_RQ_RQST!-                 ; TYPE OF MESSAGE
                                    0329   202                  <<OPC$M_NM_CENTRL@8>>          ; OPERATOR TO INFORM
                    00000000        0329   203          .LONG   0                              ; NOBODY TO RESPOND TO
52 52 45 20 2D 20 54 4D 46 52 52 45 032D   204          .ASCII  /ERRFMT - ERROR ACCESSING ERROR LOG FILE/<13><10>
47 4E 49 53 53 45 43 43 41 20 52 4F 0339
46 20 47 4F 4C 20 52 4F 52 52 45 20 0345
                  0A 0D 45 4C 49    0351
                                    0356   205 OPRMSG_END:
                                    0356   206
                                    0356   207 ROMSG:
                    00000456        0356   208          .BLKB   256                            ; HOLDS TRANSLATED STATUS MESSAGE.
                                    0456   209 ROMSG_END:
                                    0456   210 ROMSG_LEN:                                      ; HOLDS TRANSLATED MESSAGE LENGTH.
                    00000000        0456   211          .LONG   0
                                    045A   212 ;
                                    045A   213 ; MESSAGE SENT TO OPERATOR WHEN WE'VE FAILED TOO MANY TIMES TO WRITE
                                    045A   214 ; TO ERROR LOG FILE.
                                    045A   215 ;
                                    045A   216 BYEMSG_DSC:                                     ; MESSAGE DESCRIPTOR
                                    045A   217 BYEMSG_LEN:
                    00000080'       045A   218          .LONG   BYEMSG_END-BYEMSG              ; LENGTH
                    00000462'       045E   219          .LONG   BYEMSG                         ; ADDRESS
                                    0462   220
                                    0462   221 BYEMSG:                                        ; MESSAGE
                    00000103        0462   222          .LONG   OPC$_RQ_RQST! -                ; TYPE OF MESSAGE
                                    0466   223                  <<OPC$M_NM_CENTRL@8>>          ; OPERATOR TO INFORM
                    00000000        0466   224          .LONG   0                              ; NOBODY TO RESPOND TO
4C 45 44 20 2D 20 54 4D 46 52 52 45 046A   225          .ASCII  /ERRFMT - DELETING ERRFMT PROCESS/<13><10>
54 4D 46 52 52 45 20 47 4E 49 54 45 0476
      0A 0D 53 53 45 43 4F 52 50 20 0482
49 46 20 47 4F 4C 20 52 4F 52 52 45 048C   226          .ASCII  /ERROR LOG FILE UNWRITABLE/<13><10>
4C 42 41 54 49 52 57 4E 55 20 45 4C 0498
                  0A 0D 45          04A4
45 20 54 52 41 54 53 45 52 20 4F 54 04A7   227          .ASCII  /TO RESTART ERRFMT PROCESS, USE ''@SYS$SYSTEM:STARTUP ERRFMT''/
53 45 43 4F 52 50 20 54 4D 46 52 52 04B3
```

```
53 59 53 40 22 20 45 53 55 20 2C 53  04BF
52 41 54 53 3A 4D 45 54 53 59 53 24  04CB
   22 54 4D 46 52 52 45 20 50 55 54  04D7
                                     04E2    228 BYEMSG_END:
                                     04E2    229 ;
                                     04E2    230 ; MOUNT AND DISMOUNT MESSAGE STRINGS
                                     04E2    231 ;
                                     04E2    232 MOUNT_FAO:
                           00000036' 04E2    233         .LONG   MOUNT_END-MOUNT_MSG     ; LENGTH OF CONTROL STRING
                           000004EA' 04E6    234         .ADDRESS        MOUNT_MSG       ; ADDRESS OF CONTROL STRING
                                     04EA    235 MOUNT_MSG:
                           00000003  04EA    236         .LONG   OPC$_RQ_RQST            ; TYPE OF MESSAGE (OPERATOR T.B.S.)
                           00000000  04EE    237         .LONG   0                       ; NOBODY TO REPLY TO
22 44 41 21 22 20 65 6D 75 6C 6F 56  04F2    238         .ASCII  \Volume "!AD"!ASmounted, on physical device !AS\
20 2C 64 65 74 6E 75 6F 6D 53 41 21  04FE
20 6C 61 63 69 73 79 68 70 20 6E 6F  050A
      53 41 21 20 65 63 69 76 65 64  0516
                                     0520    239 MOUNT_END:
                                     0520    240
                                     0520    241 MOUNT_DSC:
                           00000080  0520    242         .LONG   128                     ; MAX SIZE OF THE MESSAGE
                           00000528' 0524    243         .ADDRESS        MOUNT_BUF       ; ADDRESS OF THE MESSAGE BUFFER
                                     0528    244 MOUNT_BUF:
                           000005A8  0528    245         .BLKB   128                     ; STORAGE FOR FORMATTED MESSAGE
                                     05A8    246 MOUNT_MNT:
              20 000005B0'010E0000'  05A8    247         .ASCID  \ \                     ; FOR VOLUME MOUNTED MESSAGE
                                     05B1    248 MOUNT_DMT:
   73 69 64 20 000005B9'010E0000'    05B1    249         .ASCID  \ dis\                  ; FOR VOLUME DISMOUNTED MESSAGE
                                     05BD    250 ;
                                     05BD    251 ; ERROR COUNTERS
                                     05BD    252 ;
                                     05BD    253 ERF$B_ERRCNT:                       ; COUNT ERRORS IN WRITING TO
                                00   05BD    254         .BYTE   0                       ; ERRORLOG FILE
                                     05BE    255 ERF$B_MAXERRCNT:                    ; MAXIMUM # ERRORS BEFORE DELETING
                                14   05BE    256         .BYTE   20                      ; THIS PROCESS
                                     05BF    257
                                     05BF    258 ;
                                     05BF    259 ; Data structures needed to get the version number and expanded file name of
                                     05BF    260 ; a newly created SYS$ERRORLOG:ERRSNAP.LOG (Venus-specific).
                                     05BF    261 ;
                                     05BF    262         .ALIGN  PAGE
                                     0600    263 ERRSNAP_FAB:
                                     0600    264         $FAB    -                       ; File Access Block.
                                     0600    265                 FNM=<SYS$ERRORLOG:ERRSNAP.LOG>, -       ; File name.
                                     0600    266                 NAM=ERRSNAP_NAM, -      ; Associated NAM block.
                                     0600    267                 XAB=ERRSNAP_XAB         ; Associated XAB block.
                                     0650    268
                                     0650    269 ERRSNAP_XAB:                        ; Declare date/time XAB.
                                     0650    270         $XABDAT
                                     067C    271
                                     067C    272 ERRSNAP_NAM:
                                     067C    273         $NAM    -                       ; Name block.
                                     067C    274                 RSA=ERRSNAP_RSA, -      ; Resultant string area address.
                                     067C    275                 RSS=NAM$C_MAXRSS        ; Use maximum length of resultant string.
                                     06DC    276
                                     06DC    277 ERRSNAP_RSA:                        ; Resultant string will be returned here.
                          000007DB   06DC    278         .BLKB   NAM$C_MAXRSS
```

```
                                07DB   279
                                07DB   280 ;
                                07DB   281 ; Data structures used when SPAWNing a sub-process to execute ERRSNAP.COM.
                                07DB   282 ;
                                07DB   283 ERRSNAP_COM:                            ; Descriptor for command procedure.
52 45 24 53 59 53 000007E3'010E0000'  07DB   284         .ASCID  /SYS$ERRORLOG:ERRSNAP.COM/
4E 53 52 52 45 3A 47 4F 4C 52 4F 52   07E9
                  4D 4F 43 2E 50 41   07F5
                                07FB   285 ERRSNAP_LOG1:                           ; Initial DCL command if copying SNAP1.
45 4C 49 46 20 24 00000803'010E0000'  07FB   286         .ASCID  /$ FILENAME := SNAP1.DAT/
50 41 4E 53 20 3D 3A 20 45 4D 41 4E   0809
                     54 41 44 2E 31   0815
                                081A   287 ERRSNAP_LOG2:                           ; Initial DCL command if copying SNAP2.
45 4C 49 46 20 24 00000822'010E0000'  081A   288         .ASCID  /$ FILENAME := SNAP2.DAT/
50 41 4E 53 20 3D 3A 20 45 4D 41 4E   0828
                     54 41 44 2E 32   0834
                                0839   289 ERRSNAP_FLAGS:                           ; Set NOCLISYM and NOWAIT flags.
                        00000006  0839   290         .LONG   6
                                083D   291 ERRSNAP_STATUS:                          ; Store the exit status of the SPAWNed
                        00000000  083D   292         .LONG   0                      ; command procedure here.
                                0841   293 ;
                                0841   294 ; Definitions needed to communicate with 11/790 logical console interface.
                                0841   295 ;
                        00000030  0841   296 CON$C_REQERL = ^X30                    ; Console command to request error
                                0841   297                                        ; snapshot file status.
                        00000031  0841   298 CON$C_INVSNP1 = ^X31                   ; Console command to invalidate SNAP1.DAT
                        00000032  0841   299 CON$C_INVSNP2 = ^X32                   ; Console command to invalidate SNAP2.DAT
                                0841   300 ERRSNAP_CONCMD:                         ; Store command to be sent to console.
                              00  0841   301         .BYTE   0
                                0842   302 ERRSNAP_DATA:                           ; Store returned data from logical
                        00000000  0842   303         .LONG   0                      ; console interface here.
                                0846   304
                                0846   305 ;
                                0846   306 ; PURE DATA - KEPT IN CODE PSECT FOR LOCALITY
                                0846   307 ;
                                0846   308
                        00000000  0000   309         .PSECT  CODE,RD,NOWRT,EXE
                                0000   310
                                0000   311 ;
                                0000   312 ; ARGUMENT LIST FOR FILE CREATE TIME STAMP ENTRY
                                0000   313 ;
                        00000001  0000   314 FILCRE:          .LONG   1              ; ONE ARGUMENT
                        00000023  0004   315                 .LONG   EMB$K_NF       ; NEW FILE TYPE MESSAGE
                                0008   316
                                0008   317 ERF$Q_DELTA:                             ; TIME BETWEEM TIME MARKS
                                0008   318
                                0008   319 ; ***    .LONG   ERF$K_CLK_TICK*ERF$K_DLTA_STMP&^X0FFFFFFFF
                                0008   320
                        9A5F4400  0008   321                 .LONG   ^X09A5F4400    ; LOW 1/2 OF DELTA TIME
                                000C   322
                                000C   323 ; ***    .LONG   ERF$K_CLK_TICK*ERF$K_DLTA_STMPa-32
                                000C   324
                        FFFFFFFE  000C   325                 .LONG   ^X0FFFFFFFE    ; HIGH 1/2 OF DELTA TIME
                                0010   326
                        FFB3B4C0  0010   327 ERF$Q_WAIT:     .LONG   -<10*1000*500> ; # OF 10 MILLISEC INTERVALS
                        FFFFFFFF  0014   328                 .LONG   -1             ; TO WAIT FOR BUFFER COMPLETION
47 4F 4C 52 4F 52 52 45 24 53 59 53   0018   329 OUTNAM: .ASCII  \SYS$ERRORLOG:ERRLOG.SYS\ ; OUTPUT FILE NAME
```

```
53 59 53 2E 47 4F 4C 52 52 45 3A  0024
                          00000017  002F   330 OUTNAMSZ = . - OUTNAM                ; LENGTH OF OUTPUT NAME
```

```
                                                .SBTTL  ERRFMT
                                        ;++
                                        ; FUNCTIONAL DESCRIPTION:
                                        ;
                                        ;       THIS PROGRAM IS AWAKENED FROM HIBERNATION BY THE ERROR LOGGER
                                        ;       WHENEVER AN ERROR LOG BUFFER BECOMES FULL.  THE ERROR FORMAT
                                        ;       PROGRAM READS THE FULL BUFFER AND THEN RELEASES IT FOR RE-USE BY
                                        ;       THE ERROR LOGGER PROGRAM.  THE DATA JUST READ IS RE-ORGANIZED
                                        ;       AND WRITTEN TO A FILE CALLED "ERRLOG.SYS" IN A FORMAT ACCEPTABLE
                                        ;       TO SYE.
                                        ;
                                        ;       THE ERROR FORMAT PROGRAM ALSO PLACES TIME STAMP ENTRIES INTO THE
                                        ;       ERROR LOG BUFFER.  THESE TIME STAMPS ARE PLACED INTO THE BUFFER
                                        ;       AT REGULAR INTERVALS.  HOWEVER, SEQUENTIAL TIME STAMPS ARE NOT
                                        ;       WRITTEN INTO THE FILE, "ERRLOG.SYS".
                                        ;
                                        ;       THE FILE, "ERRLOG.SYS", IS UPDATED, OR A NEW VERSION CREATED IF
                                        ;       THE MOST RECENT VERSION IS BEING ACCESSED OR DOES NOT EXIST.
                                        ;
                                        ;--
                                        
                                                .PSECT  CODE,RD,NOWRT,EXE
                                                .ENABL  LSB
                                        ERF$START::.ENTRY  ERF$START,0
                                                $CMKRNL_S       W^ERF$INIT      ; INITIALIZE THE ERR FORMATER
00000000'GF     04   91                        CMPB    #PR$_SID_TYP790, -      ; ARE WE EXECUTING ON A VENUS CPU?
                                                        G^EXE$GB_CPUTYPE        ;
                                                BNEQ    PRCBUF                  ; BRANCH IF NO
                                                CALLS   #0,W^ERF$ERRSNAP        ; CALL VENUS-SPECIFIC ERROR ROUTINE
                                        PRCBUF: $CMKRNL_S       W^ERF$GETBUF    ; GET THE FULL ERROR LOG BUFFER
                                                BLBS    R0,PRCNXT               ; BR IF MESSAGE(S) TO PROCESS
                                                $CLOSE  FAB=W^OUTFAB            ; CLOSE THE OUTPUT
                                                $HIBER_S                        ; WAIT FOR SOMETHING TO DO
                                                BRB     PRCBUF                  ;
                                        ;
                                        ; PROCESS NEXT MESSAGE - COME HERE WHEN A BUFFER HAS BEEN COPIED FROM
                                        ; THE SYSTEM INTO THE LOCAL BUFFER.  IF THE FILE IS NOT OPEN,
                                        ; OPEN THE OUTPUT FILE OR CREATE ONE IF MOST RECENT IS BEING ACCESSED.
                                        ;
                                        PRCNXT: CLRL    R3                      ; R3=0 => OPEN EXISTING FILE
                                                                                ; R3~=0 => CREATE NEW ERRLOG FILE
                                        PRCNXT1:
                                                MOVAB   W^INBUF,R8              ; GET ADDR OF FIRST MSG
                                                ADDB3   ERL$B_BUSY(R8),ERL$B_MSGCNT(R8),R6 ; GET COUNT OF MESSAGES
                                                BEQL    PRCBUF                  ; BR IF NO MESSAGES TO PROCESS
                                                ADDL    #ERL$C_LENGTH,R8        ; POINT TO START OF MESSAGES
                                                MOVAB   W^OUTFAB,R2             ; SET ADDRESS OF FAB
                                                TSTW    FAB$W_IFI(R2)           ; IS THE FILE OPEN?
                                                BEQL    2$                      ; BRANCH TO OPEN OR CREATE FILE
                                                BRW     NXTMSG                  ; FILE ALREADY OPEN; CONTINUE
                                        2$:
                                                CLRL    FAB$L_ALQ(R2)           ; CLEAR ALLOCATION
                                                TSTL    R3                      ; OPEN OR CREATE ERRLOG.SYS?
                                                BNEQ    5$                      ; BR TO CREATE NEW FILE
                                                $OPEN   FAB=(R2)                ; OPEN MOST RECENT VERSION
                                                BLBC    R0,4$                   ; OPEN FAILED; GO CREATE A NEW VERSION
                                        ;
```

```
                        00A1    389 ; IF THE OPEN WAS SUCCESSFUL, CHECK THAT THIS IS THE SAME ERRLOG.SYS AS THE
                        00A1    390 ; ONE WE WROTE TO LAST TIME.  IF NOT, CREATE A NEW VERSION OF ERRLOG.SYS.
                        00A1    391 ;
          02F4'CF   D5  00A1    392         TSTL    W^OUTFID                        ; HAS SYSTEM JUST RE-BOOTED?
                 2F   13  00A5    393         BEQL    10$                             ; YES; DON'T CREATE A NEW ERRLOG.SYS
       54 0294'CF   DE  00A7    394         MOVAL   W^NAMEBLOCK,R4                  ; GET ADDRESS OF NAME BLOCK
02F4'CF   24 A4   D1  00AC    395         CMPL    NAM$W_FID(R4),W^OUTFID          ; CHECK FIRST TWO WORDS OF FILE ID
                 08   12  00B2    396         BNEQ    3$                              ; FIDS DIFFER; CREATE A NEW ERRLOG.SYS
02F8'CF   28 A4   B1  00B4    397         CMPW    NAM$W_FID+4(R4),W^OUTFID+4      ; CHECK 3RD WORD OF FID
                 1A   13  00BA    398         BEQL    10$                             ; FIDS MATCH; GO CONNECT RAB
                        00BC    399 3$:
                        00BC    400         $CLOSE  FAB=(R2)                        ; CLOSE OLD FILE AND CREATE NEW ONE
                        00C5    401 4$:
                     53 D6  00C5    402         INCL    R3                              ; SIGNAL CREATING NEW FILE
                        00C7    403 5$:
                        00C7    404         $CREATE FAB=(R2)                        ; CREATE NEW VERSION
                  03 50 E8  00D0    405         BLBS    R0,10$                          ; BRANCH ON SUCCESS
                   00E5 31  00D3    406         BRW     WRITE_FAILURE                   ; NOTIFY OPERATOR OF CREATE FAILURE
    59 0250'CF   9E  00D6    407 10$:    MOVAB   W^OUTRAB,R9                     ; SET ADDRESS OF OUTPUT RAB
                  02 A9   B4  00DB    408         CLRW    RAB$W_ISI(R9)                   ; PERFORM A FAST DISCONNECT
                        00DE    409         $CONNECT RAB=(R9)                       ; CONNECT RAB TO FAB
                  03 50 E8  00E7    410         BLBS    R0,12$                          ; BRANCH ON SUCCESS
                   00CE 31  00EA    411         BRW     WRITE_FAILURE                   ; ELSE BRANCH ON FAILURE
                        00ED    412 12$:
                     53 D5  00ED    413         TSTL    R3                              ; WAS A NEW FILE JUST CREATED?
                     48 13  00EF    414         BEQL    NXTMSG                          ; BR IF NOT NEW FILE
                     53 D4  00F1    415         CLRL    R3                              ; SIGNAL SUCCESSFUL FILE CREATION
                        00F3    416                                                 ; AND INITIALIZATION
          02FA'CF   94  00F3    417         CLRB    W^LASTENTRY                     ; CLEAR SAVED MESSAGE ENTRY TYPE
       54 0294'CF   DE  00F7    418         MOVAL   W^NAMEBLOCK,R4                  ; GET ADDRESS OF NAME BLOCK
02F4'CF   24 A4   D0  00FC    419         MOVL    NAM$W_FID(R4),W^OUTFID          ; SAVE FIRST TWO WORDS OF FILE ID
02F8'CF   28 A4   B0  0102    420         MOVW    NAM$W_FID+4(R4),W^OUTFID+4      ; SAVE 3RD WORD OF FID
       5E   10   C2  0108    421         SUBL    #EMB$K_HD_LENGTH,SP             ; ALLOCATE A BUFFER (ONLY HEADER INFO)
       52   5E   D0  010B    422         MOVL    SP,R2                           ; COPY ADDRESS OF BUFFER
    28 A9   52   D0  010E    423         MOVL    R2,RAB$L_RBF(R9)                ; SET BUFFER ADDRESS IN RAB
    22 A9   10   B0  0112    424         MOVW    #EMB$K_HD_LENGTH,RAB$W_RSZ(R9) ; AND SET LENGTH FOR $PUT
 62 02FB'CF   D0  0116    425         MOVL    W^SID,EMB$L_HD_SID(R2)          ; SET SYSTEM IDENT
    04 A2   23   B0  011B    426         MOVW    #EMB$K_NF,EMB$W_HD_ENTRY(R2)    ; SET ENTRY TYPE
       0A A8   7D  011F    427         MOVQ    EMB$Q_HD_TIME+EMB$R_LENGTH(R8),- ; COPY TIME AND DATE FROM
       06 A2        0122    428                 EMB$Q_HD_TIME(R2)               ; FIRST ENTRY IN THE ERROR LOG BUFFER
       0E A2   B4   0124    429         CLRW    EMB$W_HD_ERRSEQ(R2)             ; SET ERROR SEQUENCE NUMBER OF ZERO
                        0127    430         $PUT    RAB=(R9)                        ; WRITE FILE CREATED MARK
                  03 50 E8  0130    431         BLBS    R0,15$                          ; BR IF SUCCESSFUL
                   0085 31  0133    432         BRW     WRITE_FAILURE                   ; ELSE BRANCH ON FAILURE
                        0136    433 15$:
       5E   10   C0  0136    434         ADDL    #ERF$K_TS_LENGTH,SP             ; CLEAR THE STACK
                        0139    435 ;
                        0139    436 ; PROCESS A MESSAGE IN THE ERROR BUFFER.
                        0139    437 ;
                        0139    438 ;       R6 = NUMBER OF MESSAGES IN THE BUFFER
                        0139    439 ;       R7 = IS USED TO HOLD THE FORMATTED RECORD
                        0139    440 ;       R8 = THE START OF THE NEXT MESSAGE IN THE LOCAL BUFFER
                        0139    441 ;       R9 = ADDRESS OF THE OUTPUT RAB
                        0139    442 ;
                     56 97  0139    443 NXTMSG: DECB    R6                              ; IS THERE ANOTHER MSG?
                     03 18  013B    444         BGEQ    30$                             ; BRANCH TO FORMAT ANOTHER MSG
                   FF0C 31  013D    445 20$:    BRW     PRCBUF                          ; TRY FOR ANOTHER BUFFER
```

```
                     0140   446          ASSUME  EMB$W_HD_ENTRY  EQ ERF$W_HD_ENTRY
                     0140   447          ASSUME  EMB$Q_HD_TIME   EQ ERF$Q_HD_TIME
                     0140   448          ASSUME  EMB$W_HD_ERRSEQ EQ ERF$W_HD_ERRSEQ
      58    04   C0  0140   449 30$:      ADDL    #EMB$K_LENGTH,R8          ; POINT PAST MESSAGE HEADER
    51    FC   A8  3C  0143   450          MOVZWL  EMB$W_SIZE(R8),R1         ; GET SIZE OF MESSAGE TEXT
      51    04   C2  0147   451          SUBL    #EMB$K_LENGTH,R1          ; SUBTRACT SIZE OF MESSAGE HEADER
    22 A9   51   B0  014A   452          MOVW    R1,RAB$W_RSZ(R9)          ; AND SET INTO RAB
    28 A9   68   DE  014E   453          MOVAL   (R8),RAB$L_RBF(R9)        ; AND THE ADDRESS OF THE BUFFER
       FF   A8   95  0152   454          TSTB    EMB$B_VALID(R8)           ; IS RECORD VALID?
         05      12  0155   455          BNEQ    40$                       ; BRANCH ON YES
04 A8    80   8F   88  0157   456          BISB    #ERF$M_HD_INVALD,ERF$W_HD_ENTRY(R8) ; FLAG INVALID BUFFER
    57    58   D0  015C   457 40$:      MOVL    R8,R7                     ; COPY START OF CURRENT RECORD
      58    51   C0  015F   458          ADDL    R1,R8                     ; ADVANCE TO NEXT RECORD
                     0162   459
                     0162   460          .DSABL  LSB
                     0162   461 ;
                     0162   462 ; OUTPUT ERROR MESSAGE.  R1=SIZE.
                     0162   463 ;
 26  02FA'CF    91  0162   464 MSGOUT: CMPB    W^LASTENTRY,#EMB$C_TS     ; LAST REC = TIME STAMP?
         29      12  0167   465          BNEQ    10$                       ; BRANCH ON NO
   26    04   A7  91  0169   466          CMPB    ERF$W_HD_ENTRY(R7),#EMB$C_TS ; THIS REC = TIME STAMP?
         23      12  016D   467          BNEQ    10$                       ; BRANCH ON NO
 1E A9    02   90  016F   468          MOVB    #RAB$C_RFA,RAB$B_RAC(R9)  ; SET RANDOM FILE ACCESS
                     0173   469          $FIND   RAB=(R9)                  ; FIND LAST RECORD WRITTEN
 1E A9    00   90  017C   470          MOVB    #RAB$C_SEQ,RAB$B_RAC(R9)  ; SET TO SEQUENTIAL ACCESS
       38 50   E9  0180   471          BLBC    R0,WRITE_FAILURE          ; BR IF ERROR
                     0183   472          $UPDATE RAB=(R9)                  ; UPDATE LAST RECORD
       2C 50   E9  018C   473          BLBC    R0,WRITE_FAILURE          ; BR IF ERROR
          00A7     31  018F   474          BRW     MBX                       ; BRANCH TO MAILBOX PROCESSING
02FA'CF    04   A7  90  0192   475 10$:      MOVB    ERF$W_HD_ENTRY(R7),W^LASTENTRY ; SAVE MSG ENTRY TYPE
 40 8F     04   A7  91  0198   476          CMPB    ERF$W_HD_ENTRY(R7),#EMB$C_VM  ; VOLUME MOUNTED?
         07      13  019D   477          BEQL    20$                       ; XFER IF SO
 41 8F     04   A7  91  019F   478          CMPB    ERF$W_HD_ENTRY(R7),#EMB$C_VD  ; OR VOLUME DISMOUNTED?
         09      12  01A4   479          BNEQ    30$                       ; XFER IF NOT
         57      DD  01A6   480 20$:      PUSHL   R7                        ; ELSE SAVE ADDRESS OF THE BUFFER
000003B9'EF      01  FB  01A8   481          CALLS   #1,ERF$MOUNT              ; GO FORM OPERATOR MESSAGE AND SEND IT
                     01AF   482 30$:      $PUT    RAB=(R9)                  ; OUTPUT MSG
          7E 50   E8  01B8   483          BLBS    R0,MBX                    ; BR IF SUCCESSFUL $PUT
                     01BB   484 ;
                     01BB   485 ; COME HERE IF AN ACCESS TO THE ERRORLOG FILE FAILED.
                     01BB   486 ;
                     01BB   487 WRITE_FAILURE:
                     01BB   488          $GETMSG_S -                       ; TRANSLATE REASON FOR FAILURE
                     01BB   489                  MSGID=R0, -
                     01BB   490                  MSGLEN=W^ROMSG_LEN, -
                     01BB   491                  BUFADR=W^ROMSG_DSC
    54   0315'CF    D0  01D0   492          MOVL    W^OPRMSG_LEN,R4           ; SAVE BASIC MESSAGE LENGTH
0315'CF   0456'CF    C0  01D5   493          ADDL2   W^ROMSG_LEN,W^OPRMSG_LEN; COMBINE OPRMSG WITH STATUS MSG
                     01DC   494          $SNDOPR_S -                       ; INFORM OPERATOR OF ERROR IN
                     01DC   495                  MSGBUF=W^OPRMSG_DSC       ; WRITING ERRORLOG FILE
    0315'CF    54   D0  01EA   496          MOVL    R4,W^OPRMSG_LEN           ; RESTORE BASIC MESSAGE LENGTH
                     01EF   497          $CLOSE  FAB=W^OUTFAB              ; CLOSE FILE AS CAN'T WRITE TO IT
05BD'CF 01   05BE'CF    9D  01FA   498          ACBB    W^ERF$B_MAXERRCNT,#1, - ; INC ERROR COUNT AND BRANCH IF ITS
          0010           0202   499                  W^ERF$B_ERRCNT,10$        ; <= MAX ERROR COUNT.
                     0204   500          $SNDOPR_S -                       ; ELSE NOTIFY OPERATOR THAT THIS
                     0204   501                  MSGBUF=W^BYEMSG_DSC       ; PROCESS WILL BE DELETED.
         25   11  0212   502          BRB     MBX                       ; BRANCH TO MAILBOX PROCESSING
```

```
                        0214   503 10$:                                ; ERROR COUNT <= MAX ERROR COUNT
                  53    D6   0214   504         INCL    R3             ; SIGNAL ACCESS FAILURE
          52  0200'CF    9E   0216   505         MOVAB   W^OUTFAB,R2    ; MUST CREATE NEW FILE
                02 A2    B4   021B   506         CLRW    FAB$W_IFI(R2)  ; CLEAR INDICATOR TO OPEN NEW FILE
                         021E   507         $FAB_STORE -                ; REINITIALIZE FAB
                         021E   508                 FAB=(R2), -
                         021E   509                 ORG=SEQ, -          ; SEQUENTIAL ORGANIZATION
                         021E   510                 MRS=#0, -           ; NO MAX ON RECORD SIZE
                         021E   511                 FOP=CIF, -
                         021E   512                 SHR=<GET,UPI>, -
                         021E   513                 RFM=VAR             ; VARIABLE LENGTH RECORDS
              FE39   31  0236   514         BRW     PRCNXT1             ; GO TRY TO OPEN A NEW FILE
                         0239   515 MBX:                               ; MAILBOX MESSAGES
            5B   5E    D0   0239   516         MOVL    SP,R11          ; MARK THE STACK
          50  02FF'CF   3C   023C   517         MOVZWL  W^ERF$W_MBXCHN,R0   ; MBX CHANNEL ALREADY?
                19   13   0241   518         BEQL    30$                 ; BRANCH ON NONE
0303'CF   00000000'GF   B1   0243   519         CMPW    G^EXE$GQ_ERLMBX,W^ERF$W_MBXUNT ; SAME AS LAST TIME?
                6A   13   024C   520         BEQL    50$                 ; YES, GO MAIL THE MSG
                         024E   521         $DASSGN_S       CHAN=R0     ; NO, DEASSIGN OLD CHANNEL
              02FF'CF   B4   0258   522         CLRW    W^ERF$W_MBXCHN      ; CLEAR OLD CHANNEL
                         025C   523 30$:
          50  00000000'GF   3C   025C   524         MOVZWL  G^EXE$GQ_ERLMBX,R0  ; GET NEW MAIL BOX UNIT
            0303'CF   50   B0   0263   525         MOVW    R0,W^ERF$W_MBXUNT   ; SET NEW UNIT TO USE
                2A   13   0268   526         BEQL    40$                 ; BRANCH IF NONE
              5E   1C   C2   026A   527         SUBL    #32-4,SP            ; ALLOCATE BUFFER IN THE STACK
              52   5E   D0   026D   528         MOVL    SP,R2               ; MARK START OF MAIL BOX UNIT
       41424D5F   8F   DD   0270   529         PUSHL   #^A/_MBA/           ; SET PROTOTYPE NAME
              5E   DD   0276   530         PUSHL   SP                  ; SET START OF BUFFER
            00C3   30   0278   531         BSBW    100$                ; SET UNIT OF MAILBOX
        7E  52   6E   C3   027B   532         SUBL3   (SP),R2,-(SP)       ; FIND LENGTH OF NAME
            52   5E   D0   027F   533         MOVL    SP,R2               ; SAVE POINTER TO NAME
                         0282   534         $ASSIGN_S       DEVNAM=(R2),-   ; ASSIGN A CHANNEL TO
                         0282   535                         CHAN=W^ERF$W_MBXCHN; THE DIAGNOSTIC MAILBOX
            03 50    E8   0291   536         BLBS    R0,45$              ; BRANCH ON SUCCESS
             008D    31   0294   537 40$:     BRW     65$                 ; SKIP THE QIO IF FAILED
                         0297   538 45$:
          6E   20    D0   0297   539         MOVL    #32,(SP)            ; RESET LENGTH OF BUFFER
                         029A   540         $GETCHN_S       CHAN=W^ERF$W_MBXCHN,-; GET SIZE OF MAILBOX
                         029A   541                         PRIBUF=(R2)     ; I.E., THE MAXIMUM MSG SIZE
        52   04 A2    D0   02AE   542         MOVL    4(R2),R2            ; GET ADDRESS OF DEV CHAR BUFFER
  0301'CF    06 A2    B0   02B2   543         MOVW    DIB$W_DEVBUFSIZ(R2),W^ERF$W_MBXSIZ ; GET MAILBOX SIZE
        50   22 A9    3C   02B8   544 50$:     MOVZWL  RAB$W_RSZ(R9),R0    ; GET SIZE OF MESSAGE
     0301'CF   50   B1   02BC   545         CMPW    R0,W^ERF$W_MBXSIZ   ; MSG TOO LARGE?
               05   1B   02C1   546         BLEQU   55$                 ; BRANCH ON OK
     50   0301'CF   B0   02C3   547         MOVW    W^ERF$W_MBXSIZ,R0   ; TRUNCATE MSG
                         02C8   548 55$:     $QIO_S  CHAN=W^ERF$W_MBXCHN,-   ; CHANNEL FOR DIAG MBX
                         02C8   549                 FUNC=#<IO$_WRITEVBLK!IO$M_NOW>,- ; DONT WAIT FOR SUCCESS
                         02C8   550                 P1=(R7),-           ; ADDR OF ERROR MSG
                         02C8   551                 P2=R0               ; SIZE OF MSG
05BE'CF   05BD'CF   91   02E7   552         CMPB    W^ERF$B_ERRCNT, -   ; HAVE WE EXCEEDED THE ERROR
                         02EE   553                 W^ERF$B_MAXERRCNT   ; THRESHHOLD?
               34   15   02EE   554         BLEQ    65$                 ; BRANCH IF NO
        50   045A'CF   3C   02F0   555         MOVZWL  W^BYEMSG_LEN,R0     ; GET LENGTH OF GOODBYE MESSAGE
00000301'EF     50   B1   02F5   556         CMPW    R0,ERF$W_MBXSIZ     ; MESSAGE TOO LARGE?
               05   1B   02FC   557         BLEQU   60$                 ; BRANCH ON OK
     50   0301'CF   B0   02FE   558         MOVW    W^ERF$W_MBXSIZ,R0   ; TRUNCATE MESSAGE
                         0303   559 60$:
```

```
                                0303   560          $QIO_S -                                ; NOTIFY MAILBOX THAT PROCESS IS
                                0303   561                 CHAN=W^ERF$W_MBXCHN, -           ; BEING DELETED.
                                0303   562                 FUNC=#<IO$_WRITEVBLK!IO$M_NOW>, -
                                0303   563                 P1=W^BYEMSG, -
                                0303   564                 P2=R0
                          5E    5B  D0  0324   565 65$:     MOVL    R11,SP                  ; RESET THE STACK POINTER
          05BE'CF       05BD'CF   91  0327   566          CMPB    W^ERF$B_ERRCNT, -       ; HAVE WE EXCEEDED THE ERROR
                                      032E   567                  W^ERF$B_MAXERRCNT       ; THRESHHOLD?
                             03   14  032E   568          BGTR    70$                     ; BRANCH IF YES
                           FE06   31  0330   569          BRW     NXTMSG                  ; ELSE GO PROCESS NEXT MESSAGE
                                0333   570 ;
                                0333   571 ; IF ERRCNT > MAXERRCNT, DELETE THIS PROCESS TO PREVENT INFINITE LOOPING.
                                0333   572 ; THE ERRFMT PROCESS CAN BE RESTARTED VIA AN OPERATOR COMMAND FILE.
                                0333   573 ;
                                0333   574 70$:     $DELPRC_S                               ; DELETE THIS PROCESS
                                033E   575
                                033E   576 ;
                                033E   577 ; LOCAL SUBROUTINE TO CONVERT BINARY TO ASCII AND STORE RESULT
                                033E   578 ; IN BUFFER POINTED TO BY R2
                                033E   579 ;
                                033E   580
                             51    D4  033E   581 100$:    CLRL    R1                      ; ZERO HI 1/2 OF QUAD WORD
          7E    50       50    0A  7B  0340   582 110$:    EDIV    #10,R0,R0,-(SP)         ; GET NEXT DIGIT
                         6E    30  C0  0345   583          ADDL    #^A/0/,(SP)             ; FIND THE DIGIT IN ASCII
                             50    D5  0348   584          TSTL    R0                      ; ANY THING LEFT
                             02    13  034A   585          BEQL    120$                    ; BR IF NO MORE TO CONVERT
                             F2    10  034C   586          BSBB    110$                    ; GET NEXT DIGIT
                         82    8E    F6  034E   587 120$:    CVTLB   (SP)+,(R2)+             ; STORE A BYTE
                                     05  0351   588          RSB                             ;
```

```
                                0352   590          .SBTTL  ERRFMT KERNAL MODE INIT
                                0352   591 ;++
                                0352   592 ;
                                0352   593 ; ERF$INIT - INITIALIZE THE ERROR FORMAT PROGRAM
                                0352   594 ;
                                0352   595 ; THIS ROUTINE IS ENTERED AT KERNAL MODE TO DELETE A PRVIOUS COPY
                                0352   596 ; OF THIS PROCESS IF ONE EXISTS, TEHREBY PERMITTING ONLINE REPLACEMENT
                                0352   597 ; OF THE ERROR FORMAT PROGRAM, AS WELL AS EASE OF TESTING. ALSO, THE
                                0352   598 ; KERNAL MODE TIMER AST FOR TIME STAMPING THE ERROR LOG IS STARTED.
                                0352   599 ;--
                                0352   600 ;
                                0352   601          .ENABL  LSB
                          000C  0352   602          .ENTRY  ERF$INIT,^M<R2,R3>      ;
 52   00000000'EF      DE      0354   603          MOVAL   ERL$GL_ERLPID,R2        ; GET ADDRESS OF CURRENT PID
 53   00000000'GF      D0      035B   604          MOVL    G^SCH$GL_CURPCB,R3      ; GET CURRENT PCB
                62    D5      0362   605          TSTL    (R2)                    ; PID EGUAL ZERO?
                1D    13      0364   606          BEQL    10$                     ; BR IF YES - NO PREVIOUS ERRFMT
          62    60    A3      D1      0366   607          CMPL    PCB$L_PID(R3),(R2)      ; MAKE SURE IT IS THIS PROCESS
                17    13      036A   608          BEQL    10$                     ; BR IF SAME PROCESS
             50 62    D0      036C   609          MOVL    (R2),R0                 ; GET INTERNAL PID TO R0
          00000000'GF 16      036F   610          JSB     G^EXE$IPID_TO_EPID      ; CONVERT TO EXTENDED PID
             62 50    D0      0375   611          MOVL    R0,(R2)                 ; SAVE IT IN THE SAME PLACE FOR DELPRC
                                0378   612          $DELPRC_S PIDADR=(R2)           ; DELETE OLD ERRFMT
          62    60    A3      D0      0383   613 10$:     MOVL    PCB$L_PID(R3),(R2)      ; SET THE PID FOR THIS PROCESS
       02FB'CF      3E      DB      0387   614          MFPR    #PR$_SID,W^SID          ; GET SYS ID REGISTER
                18    11      038C   615          BRB     30$                     ;
```

```
                        038E   617          .SBTTL  TIME STAMP ROUTINE
                        038E   618
                        038E   619 ;++
                        038E   620 ;
                        038E   621 ; ERF$TIMSTMP - TIME STAMP
                        038E   622 ;
                        038E   623 ; THIS ROUTINE IS ENTERED PERIODICALLY TO ENTER A TIME STAMP INTO
                        038E   624 ; THE ERROR MESSAGE BUFFER.  THEY ARE REMOVED ALONG WITH ANY OTHER
                        038E   625 ; ENTRIES MADE BY THE MAIN LINE OF THIS PROGRAM.
                        038E   626 ;
                        038E   627 ; THIS ROUTINE HAS NO INPUTS AND ONLY OUTPUT IS THE ENTRY OF THE
                        038E   628 ; TIME STAMP IN THE ERROR LOG BUFFER.  IF A BUFFER CAN NOT BE
                        038E   629 ; ALLOCATED, THAT TIME STAMP IS LOST.
                        038E   630 ;--
                        038E   631
                  0004  038E   632          .ENTRY  ERF$TIMSTMP,^M<R2>              ; TIME STAMP ROUTINE
       51    10   9A    0390   633          MOVZBL  #EMB$C_TS_LENGTH,R1     ; GET LENGTH OF MSG
00000000'EF     16    0393   634          JSB     ERL$ALLOCEMB            ; GO GET A MSG BLOCK
         0A     50    E9    0399   635          BLBC    R0,30$                  ; BRANCH ON NO BLOCK
   04    A2     26    B0    039C   636          MOVW    #EMB$K_TS,EMB$W_HD_ENTRY(R2) ; SET ENTRY TYPE
00000000'EF     16    03A0   637          JSB     ERL$RELEASEMB           ; RELEASE BLOCK
                        03A6   638 30$:     $SETIMR_S       -               ; SET A TIMER FOR TIME STAMPS
                        03A6   639                      DAYTIM=ERF$Q_DELTA,-    ; TIMER DELTA TIME
                        03A6   640                      ASTADR=ERF$TIMSTMP,-    ; THE TIME STAMP ENTRY
                        03A6   641                      REQIDT=#EMB$K_TS        ; AST PARAMETER IS MESSAGE TYPE
                     04 03B8   642          RET
                        03B9   643
                        03B9   644          .DSABL  LSB
```

```
                        03B9    646          .SBTTL  VOLUME MOUNT/DISMOUNT MESSAGE ROUTINE
                        03B9    647
                        03B9    648 ;++
                        03B9    649 ;
                        03B9    650 ; ERF$MOUNT - MOUNT STATUS MESSAGE
                        03B9    651 ;
                        03B9    652 ; THIS ROUTINE IS PASSED THE ADDRESS OF THE MESSAGE. FROM THE MESSAGE, IT
                        03B9    653 ; BUILDS A MESSAGE FOR THE OPERATOR INDICATING THE MOUNT STATUS (MOUNTED OR
                        03B9    654 ; DISMOUNTED) AND SENDS IT TO THE APPROPRIATE OPERATOR. TAPE MOUNTS GO TO
                        03B9    655 ; THE 'TAPES' OPERATOR, DISK MOUNTS GO TO THE 'DISKS' OPERATOR, AND ANY OTHER
                        03B9    656 ; MESSAGES GO TO THE 'DEVICES' OPERATOR FOR HANDLING.
                        03B9    657 ;
                        03B9    658 ;--
                        03B9    659
                077C    03B9    660          .ENTRY  ERF$MOUNT,^M<R2,R3,R4,R5,R6,R8,R9,R10>
                        03BB    661 ;
                        03BB    662 ; DETERMINE IF A MESSAGE SHOULD BE SENT.
    51  00000000'GF  D0 03BB    663          MOVL    G^EXE$GL_MSGFLAGS,R1            ; GET SYSTEM MESSAGE FLAGS
       40 8F   04 A7  91 03C2    664          CMPB    EMB$W_HD_ENTRY(R7),#EMB$C_VM    ; IS THIS A MOUNT NOTIFICATION?
                   09  12 03C7    665          BNEQ    20$                             ; BRANCH IF NOT
09  51  00000000'8F  E0 03C9    666          BBS     #EXE$V_MOUNTMSG,R1,SNDMSG       ; BR IF MOUNT NOTIFICATION DESIRED
                   04  03D1    667 10$:     RET                                     ; OTHERWISE RETURN (NO STATUS)
F7  51  00000000'8F  E1 03D2    668 20$:     BBC     #EXE$V_DISMOUMSG,R1,10$         ; BR IF DISMOUNT NOTIFICATION NOT DE
                        03DA    669
                        03DA    670 ;
                        03DA    671 ; BUILD A BUFFER DESCRIPTOR AND USE IT TO HOLD THE FORMATTED DEVICE NAME.
                        03DA    672 ;
          5E   1C   C2  03DA    673 SNDMSG: SUBL2   #28,SP                          ; MAKE ROOM FOR DESCRIPTOR AND BUFFER
          56   5E   D0  03DD    674          MOVL    SP,R6                          ; COPY DESCRIPTOR ADDRESS
          86   14   D0  03E0    675          MOVL    #20,(R6)+                      ; SET DEVICE NAME BUFFER LENGTH
       86   04 A6   DE  03E3    676          MOVAL   4(R6),(R6)+                    ; SET DEVICE NAME BUFFER ADDRESS
          56   5E   D0  03E7    677          MOVL    SP,R6                          ; RESET DESCRIPTOR ADDRESS
       58   1E A7   9E  03EA    678          MOVAB   ERF$B_VM_NAMLNG(R7),R8         ; GET ADDRESS OF DEVICE ASCIC STRING
       59   1C A7   3C  03EE    679          MOVZWL  ERF$W_VM_UNIT(R7),R9           ; GET DEVICE UNIT NUMBER
                        03F2    680          $FAO_S  DEVFAO,(R6),(R6),R8,R9         ; FORMAT THE DEVICE NAME
                        0407    681 ;
                        0407    682 ; BUILD A $GETDVI ITEM LIST ON THE STACK AND
                        0407    683 ; CALL $GETDVI TO DETERMINE THE DEVICE CLASS.
                        0407    684 ;
             7E   7C    0407    685          CLRQ    -(SP)                          ; MAKE ROOM ON THE STACK FOR THE
             7E   7C    0409    686          CLRQ    -(SP)                          ; $GETDVI ITEM LIST
          58 5E   D0    040B    687          MOVL    SP,R8                          ; SAVE ADDRESS FOR LATER
68  00040004 8F   D0    040E    688          MOVL    #<DVI$_DEVCLASS@16>!4,(R8)     ; SET ITEM CODE AND BUFFER SIZE
             7E   D4    0415    689          CLRL    -(SP)                          ; MAKE ROOM FOR THE DEVICE CLASS
       04 A8 5E   D0    0417    690          MOVL    SP,4(R8)                       ; NOTE THE STORAGE ADDRESS
                        041B    691                                                 ; NO RETURN LENGTH NEEDED
                        041B    692          $GETDVI_S       EFN=#6,-               ; GET THE NEEDED DEVICE INFO
                        041B    693                          DEVNAM=(R6),-
                        041B    694                          ITMLST=(R8)
                        0431    695          $WAITFR_S       EFN=#6                 ; WAIT UNTIL COMPLETE
             51 8ED0    043A    696          POPL    R1                             ; GET THE DEVICE CLASS
    50  0803 8F   3C    043D    697          MOVZWL  #<OPC$M_NM_DISKS@8>!OPC$_RQ_RQST,R0 ; SET FOR DISK OPERATOR
          01   51   91  0442    698          CMPB    R1,#DC$_DISK                   ; WAS IT A DISK DEVICE?
               0F   13  0445    699          BEQL    10$                            ; XFER IF SO
    50  0403 8F   3C    0447    700          MOVZWL  #<OPC$M_NM_TAPES@8>!OPC$_RQ_RQST,R0 ; ELSE SET FOR TAPE
          02   51   91  044C    701          CMPB    R1,#DC$_TAPE                   ; WAS IT A TAPE DEVICE?
               05   13  044F    702          BEQL    10$                            ; XFER IF SO
```

```
                50    1003 8F   3C   0451    703                MOVZWL  #<OPC$M_NM_DEVICE@8>!OPC$_RQ_RQST,R0    ; ELSE UNKNOWN.
        000004EA'EF     50   D0   0456    704 10$:           MOVL    R0,MOUNT_MSG                ; SET OPERATOR NAME
                                   045D    705 ;
                                   045D    706 ; FORMAT THE OPERATOR MESSAGE AND SEND IT TO OPCOM.
                                   045D    707 ;
    00000520'EF    0080 8F   B0   045D    708                MOVW    #128,MOUNT_DSC              ; RESET DESCRIPTOR SIZE
                     56   DD   0466    709                PUSHL   R6                          ; SET DEVICE NAME DESCRIPTOR
              000005A8'EF   9F   0468    710                PUSHAB  MOUNT_MNT                   ; SET FOR MOUNT MESSAGE
            40 8F    04   A7   91   046E    711                CMPB    EMB$W_HD_ENTRY(R7),#EMB$C_VM    ; RIGHT?
                     07   13   0473    712                BEQL    40$                         ; XFER IF SO
          6E    000005B1'EF   9E   0475    713                MOVAB   MOUNT_DMT,(SP)              ; ELSE SET FOR DISMOUNT MESSAGE
                     32   A7   9F   047C    714 40$:           PUSHAB  ERF$T_VM_LABEL(R7)          ; ADDRESS OF VOLUME LABEL
                     0C   DD   047F    715                PUSHL   #12                         ; SIZE OF THE LABEL
                                   0481    716                $FAO_S  MOUNT_FAO,MOUNT_DSC,MOUNT_DSC   ;FORMAT THE MESSAGE
                                   049A    717                $SNDOPR_S       MSGBUF=MOUNT_DSC        ; SEND THE MESSAGE
                          04   04AA    718                RET                                 ; RETURN WHEN DONE
```

```
                                04AB   720          .SBTTL  GET ERROR LOG BUFFER
                                04AB   721 ;++
                                04AB   722 ;
                                04AB   723 ; ERF$GETBUF - GET ERROR LOG BUFFER
                                04AB   724 ;
                                04AB   725 ; THIS ROUTINE IS CALLED IN KERNAL MODE TO GET A BUFFER OF ERROR
                                04AB   726 ; MESSAGES FORM THE ERROR LOG FACILITY. IF THE BUFFER HAS BUSY
                                04AB   727 ; MESSAGES, THIS PROCESS WAITS FOR A WHILE.  IF THE MESSAGE DOES
                                04AB   728 ; NOT GO UNBUSY IN A REASONABLE TIME, THE BUFFER IS TAKEN IN ITS
                                04AB   729 ; INDETERMINATE FORM.
                                04AB   730 ;
                                04AB   731 ; RETURN OF R0 = FALSE INDICATES NO BUFFERS WERE READY,
                                04AB   732 ;       TRUE INDICATES A BUFFER WAS OBTAINED.
                                04AB   733 ;--
                          04FC  04AB   734          .ENTRY  ERF$GETBUF,^M<R2,R3,R4,R5,R6,R7,R10>; ENTRY POINT MASK
          5A    FF 8F     9A  04AD   735          MOVZBL  #ERF$C_LOOP_CNT,R10     ; SET A LOOP COUNT
    54  00000000'EF      9A  04B1   736          MOVZBL  ERL$GB_BUFPTR,R4        ; GET BUFFER POINTER
56  00000000'EF44       D0  04B8   737          MOVL    ERL$AL_BUFADDR[R4],R6   ; GET BUFFER ADDRESS
    57    0000'CF       9E  04C0   738          MOVAB   W^INBUF,R7              ; GET ADDR OF STORAGE BUF
    00 03 A6    00      E6  04C5   739          BBSSI   #ERL$V_LOCK,ERL$B_FLAGS(R6),20$; INHIBIT ALLOCATIONS
                  66      95  04CA   740 20$:    TSTB    ERL$B_BUSY(R6)          ; IS BUFFER CHANGING?
                  1E      13  04CC   741          BEQL    30$                     ; BR IF NO
                  5A      D7  04CE   742          DECL    R10                     ; IS WAIT TIME UP FOR THIS BUFFER?
                  1A      13  04D0   743          BEQL    30$                     ; BR IF YES
                              04D2   744          $SETIMR_S       #2,ERF$Q_WAIT   ; WAIT FOR A BIT
                              04E1   745          $WAITFR_S       #2              ; FOR THE MESSAGES TO COMPLETE
                  DE      11  04EA   746          BRB     20$                     ; CHECK THE BUFFER AGAIN
          5A    FF 8F     9A  04EC   747 30$:    MOVZBL  #ERF$C_LOOP_CNT,R10     ; SET A LOOP COUNT
67  66    0200 8F     28  04F0   748 35$:    MOVC3   #512,(R6),(R7)          ; COPY BUFFER
67  66    0200 8F     29  04F6   749          CMPC3   #512,(R6),(R7)          ; DID BUFFER CHANGE ?
                  1E      13  04FC   750          BEQL    40$                     ; IF EQL, OK
                  5A      D7  04FE   751          DECL    R10                     ; IS WAIT TIME UP FOR THIS BUFFER?
                  1A      13  0500   752          BEQL    40$                     ; BR IF YES
                              0502   753          $SETIMR_S       #2,ERF$Q_WAIT   ; WAIT FOR A BIT
                              0511   754          $WAITFR_S       #2              ; FOR THE MESSAGES TO COMPLETE
                  D4      11  051A   755          BRB     35$                     ; CHECK THE BUFFER AGAIN
                              051C   756 40$:    DSBINT                          ; DISABLE INTERRUPTS
                  66      B4  0522   757          CLRW    ERL$B_BUSY(R6)          ; CLEAR MESSAGE AND BUSY COUNTS
    08 A6    04 A6      D1  0524   758          CMPL    ERL$L_NEXT(R6),ERL$L_END(R6) ; WAS BUFFER FULL?
                  07      12  0529   759          BNEQU   50$                     ; IF NEQU NO
00000000'EF       01      8C  052B   760          XORB    #1,ERL$GB_BUFPTR        ; INDICATE NEXT BUFFER TO READ
    04 A6    0C A6      9E  0532   761 50$:    MOVAB   ERL$C_LENGTH(R6),ERL$L_NEXT(R6) ; SET ALL BUFFER FREE
    00 03 A6    00      E7  0537   762          BBCCI   #ERL$V_LOCK,ERL$B_FLAGS(R6),60$; ENABLE ALLOCATIONS
                              053C   763 60$:    ENBINT                          ; ENABLE INTERRUPTS
     50   0001'CF      9A  053F   764          MOVZBL  W^INBUF+ERL$B_MSGCNT,R0 ; ANY COMPLETED MESSAGES?
                  03      13  0544   765          BEQL    70$                     ; IF EQL NO MESSAGES
          50      01      D0  0546   766          MOVL    #1,R0                   ; SET SUCESSFUL INDICATION
                  04      05  0549   767 70$:    RET                             ; RETURN
```

```
                     054A   769          .SBTTL  ERF$ERRSNAP
                     054A   770 ;++
                     054A   771 ; FUNCTIONAL DESCRIPTION:
                     054A   772 ;       This routine is specific to the VENUS CPU.  It allows a VENUS-specific
                     054A   773 ;       errorlog to be copied from the VENUS console mass-storage device
                     054A   774 ;       to the SYS$ERRORLOG area, once per bootstrap.
                     054A   775 ;
                     054A   776 ; CALLING SEQUENCE:
                     054A   777 ;
                     054A   778 ;       CALLS/G #0,ERF$ERRSNAP
                     054A   779 ;
                     054A   780 ; INPUT PARAMETERS:
                     054A   781 ;
                     054A   782 ;       NONE
                     054A   783 ;
                     054A   784 ; IMPLICIT INPUTS:
                     054A   785 ;
                     054A   786 ;       The VENUS console subsystem will be queried to find if a valid error
                     054A   787 ;       snapshot file exists on the venus console mass storage device.
                     054A   788 ;
                     054A   789 ; OUTPUT PARAMETERS:
                     054A   790 ;
                     054A   791 ;       NONE
                     054A   792 ;
                     054A   793 ; IMPLICIT OUTPUTS:
                     054A   794 ;
                     054A   795 ;       If the error snapshot file exists on the venus console mass storage
                     054A   796 ;       device, it will be copied to the SYS$ERRORLOG directory.
                     054A   797 ;
                     054A   798 ; COMPLETION CODES:
                     054A   799 ;
                     054A   800 ;       NONE
                     054A   801 ;
                     054A   802 ; SIDE EFFECTS:
                     054A   803 ;
                     054A   804 ;       NONE
                     054A   805 ;
                     054A   806 ;--
                     054A   807
                0000 054A   808          .ENTRY  ERF$ERRSNAP,0
                     054C   809 ;
                     054C   810 ; Determine which, if either, snapshot file is present and valid on the
                     054C   811 ; console device.
                     054C   812 ;
          0842'CF    D4  054C   813          CLRL    W^ERRSNAP_DATA          ; Initialize returned data buffer.
                     0550   814          $CMKRNL_S -                     ; Is a snapshot log present?
                     0550   815                  ERF$SNAPSHOT_PRESENT
0842'CF   0843'CF    90  055F   816          MOVB    W^ERRSNAP_DATA+1, -     ; Copy the flag byte to the low
                     0566   817                  W^ERRSNAP_DATA          ; byte of the buffer.
          0842'CF    95  0566   818 10$:     TSTB    W^ERRSNAP_DATA          ; Is SNAP1 or SNAP2 valid?
                67   13  056A   819          BEQL    RETURN_STATUS           ; Branch if neither.
     52   07FB'CF    9E  056C   820          MOVAB   W^ERRSNAP_LOG1,R2       ; Assume SNAP1 is valid.
                31   90  0571   821          MOVB    #CON$C_INVSNP1,-        ; Save function code to invalidate
          0841'CF        0573   822                  W^ERRSNAP_CONCMD        ; SNAP1 file when we're done with it.
  10 0842'CF    00   E4  0576   823          BBSC    #0,W^ERRSNAP_DATA,20$   ; Branch if SNAP1 is valid.
     52   081A'CF    9E  057C   824          MOVAB   W^ERRSNAP_LOG2,R2       ; SNAP2 must be valid.
                32   90  0581   825          MOVB    #CON$C_INVSNP2,-        ; Save function code to invalidate
```

```
                0841'CF      0583   826                 W^ERRSNAP_CONCMD        ; SNAP2 file when we're done with it.
          0842'CF   01  E5   0586   827         BBCC    #1,W^ERRSNAP_DATA,-     ; Branch only if invalid data from the
                    47       058B   828                 RETURN_STATUS           ; console logical interface.
                             058C   829 20$:
                             058C   830 ;
                             058C   831 ; SPAWN a subprocess to execute the ERRSNAP.COM command procedure.
                             058C   832 ; ERRSNAP.COM copies the error snapshot file from the console device to
                             058C   833 ; SYS$ERRORLOG:ERRSNAP.LOG.
                             058C   834 ;
                0839'CF      DF     058C   835         PUSHAL  W^ERRSNAP_STATUS        ; To receive completion status from the
                             0590   836                                                 ; command procedure.
                      00     DD     0590   837         PUSHL   #0                      ; No process-id.
                      00     DD     0592   838         PUSHL   #0                      ; Don't care what the process' name is.
                0839'CF      DF     0594   839         PUSHAL  W^ERRSNAP_FLAGS         ; Specify NOCLISYM and NOLOGNAM for flags.
                      00     DD     0598   840         PUSHL   #0                      ; Use caller's SYS$OUTPUT.
                07DB'CF      7F     059A   841         PUSHAQ  W^ERRSNAP_COM           ; Define ERRSNAP.COM as SYS$INPUT.
                      52     DD     059E   842         PUSHL   R2                      ; Address of initial command string.
00000000'GF           07     FB     05A0   843         CALLS   #7,G^LIB$SPAWN          ; Execute the command procedure.
                   29 50     E9     05A7   844         BLBC    R0,RETURN_STATUS        ; Check status of LIB$SPAWN.
50    0000083D'EF     D0     05AA   845         MOVL    ERRSNAP_STATUS,R0       ; Check status returned by command
                   1F 50     E9     05B1   846         BLBC    R0,RETURN_STATUS        ; procedure.
                             05B4   847 ;
                             05B4   848 ; Notify the console interface that we copied the file successfully, and put
                             05B4   849 ; some information about the newly created file into ERRLOG.SYS.
                             05B4   850 ;
                             05B4   851         $OPEN   FAB=W^ERRSNAP_FAB
                   11 50     E9     05BF   852         BLBC    R0,RETURN_STATUS
                             05C2   853         $CMKRNL_S -
                             05C2   854                 ERF$SNAPSHOT_COPIED
                      93     11     05D1   855         BRB     10$                     ; Make sure we get both files if both
                             05D3   856                                                 ; contain valid information.
                             05D3   857 RETURN_STATUS:
                         04  05D3   858         RET
```

```
                     05D4   860          .SBTTL  ERF$SNAPSHOT_PRESENT
                     05D4   861 ;++
                     05D4   862 ; FUNCTIONAL DESCRIPTION:
                     05D4   863 ;       Query the VENUS console to find if a valid error snapshot log is
                     05D4   864 ;       present on the console mass storage device.
                     05D4   865 ;
                     05D4   866 ; CALLING SEQUENCE:
                     05D4   867 ;
                     05D4   868 ;       $CMKRNL_x       ERF$SNAPSHOT_PRESENT
                     05D4   869 ;
                     05D4   870 ; INPUT PARAMETERS:
                     05D4   871 ;
                     05D4   872 ;       NONE
                     05D4   873 ;
                     05D4   874 ; OUTPUT PARAMETERS:
                     05D4   875 ;
                     05D4   876 ;       R0      - Low bit set means no errors encoutered
                     05D4   877 ;       ERRSNAP_DATA:
                     05D4   878 ;           byte 0:
                     05D4   879 ;               - contains a ^x20
                     05D4   880 ;           byte 1:
                     05D4   881 ;               - bit 0 is set if SNAP1.DAT is valid on the console disk
                     05D4   882 ;               - bit 1 is set if SNAP2.DAT is valid on the console disk
                     05D4   883 ;               - bits <7:2> MBZ
                     05D4   884 ;
                     05D4   885 ;--
                000C 05D4   886          .ENTRY  ERF$SNAPSHOT_PRESENT,^M<R2,R3>
                     05D6   887 ;
                     05D6   888 ; Call CON$SENDCONSCMD to determine if there is a valid error snapshot on the
                     05D6   889 ; console disk.
                     05D6   890 ;
      50    30   D0 05D6   891          MOVL    #CON$C_REQERL,R0        ; Function = request errorlog status.
      52    02   D0 05D9   892          MOVL    #2,R2                   ; Expect 2 bytes of returned data.
53  0842'CF    9E 05DC   893          MOVAB   W^ERRSNAP_DATA,R3       ; Address of buffer to return data in.
 00000000'GF    16 05E1   894          JSB     G^CON$SENDCONSCMD       ; Send command to logical console.
                04 05E7   895 10$:     RET
```

```
                                05E8    897          .SBTTL  ERF$SNAPSHOT_COPIED
                                05E8    898 ;++
                                05E8    899 ; FUNCTIONAL DESCRIPTION:
                                05E8    900 ;       This routine is called after the VENUS error snapshot has been copied to
                                05E8    901 ;       the SYS$ERRORLOG directory.  It signals the console interface to
                                05E8    902 ;       invalidate the contents of the error snapshot container file on the
                                05E8    903 ;       VENUS console mass storage device.
                                05E8    904 ;
                                05E8    905 ; CALLING SEQUENCE:
                                05E8    906 ;
                                05E8    907 ;       $CMKRNL ERF$SNAPSHOT_COPIED
                                05E8    908 ;
                                05E8    909 ; INPUT PARAMETERS:
                                05E8    910 ;
                                05E8    911 ;       NONE
                                05E8    912 ;
                                05E8    913 ; OUTPUT PARAMETERS:
                                05E8    914 ;
                                05E8    915 ;       NONE
                                05E8    916 ;
                                05E8    917 ;--
                        003C    05E8    918          .ENTRY  ERF$SNAPSHOT_COPIED,^M<R2,R3,R4,R5>
                                05EA    919 ;
                                05EA    920 ; Write an entry into the error log indicating that a new ERRSNAP.LOG file
                                05EA    921 ; has been created.  Include its fully expanded name, creation date, and its
                                05EA    922 ; file-id.
                                05EA    923 ;
                53   067F'CF    9A   05EA    924          MOVZBL  W^ERRSNAP_NAM+NAM$B_RSL,R3; Get size of resultant file name.
                51   1E   53    C1   05EF    925          ADDL3   R3,#<EMB$C_HD_LENGTH+8+6>, -; Calculate size of buffer to
                                05F3    926                  R1                          ; allocate.
                00000000'GF     16   05F3    927          JSB     G^ERL$ALLOCEMB              ; Allocate an error message buffer.
                      37 50     E9   05F9    928          BLBC    R0,10$                      ; Branch if failed to get a buffer.
                 04 A2    10    B0   05FC    929          MOVW    #EMB$C_HLT, -               ; Store the error entry type.
                                0600    930                  EMB$W_HD_ENTRY(R2)
                    06A0'CF     D0   0600    931          MOVL    W^ERRSNAP_NAM+NAM$W_FID,-; Put the first two words of the
                      10 A2          0604    932                  EMB$C_HD_LENGTH(R2)     ; file-id in the buffer.
                    06A4'CF     B0   0606    933          MOVW    W^ERRSNAP_NAM+NAM$W_FID+4,-; And the last word of the file-id.
                      14 A2          060A    934                  EMB$C_HD_LENGTH+4(R2)
                    0664'CF     7D   060C    935          MOVQ    W^ERRSNAP_XAB+XAB$Q_CDT,-; Put the file's creation date and
                      16 A2          0610    936                  EMB$C_HD_LENGTH+6(R2)   ; time into the buffer.
                           06   BB   0612    937          PUSHR   #^M<R1,R2>
1E A2           06DC'CF     53   28   0614    938          MOVC3   R3,W^ERRSNAP_RSA, -         ; Move the resultant file name into
                                061B    939                  EMB$C_HD_LENGTH+14(R2)  ; the error log buffer.
                           06   BA   061B    940          POPR    #^M<R1,R2>
                00000000'GF     16   061D    941          JSB     G^ERL$RELEASEMB             ; Release the errorlog data.
                                0623    942 ;
                                0623    943 ; Now notify the console interface that we have copied the error snapshot
                                0623    944 ; file successfully.
                                0623    945 ;
                50   0841'CF    9A   0623    946          MOVZBL  W^ERRSNAP_CONCMD, -         ; Get console command to invalidate
                                0628    947                  R0                          ; correct snapshot file.
                           52   D4   0628    948          CLRL    R2                          ; Not expecting any returned data.
                00000000'GF     16   062A    949          JSB     G^CON$SENDCONSCMD           ; Send command to the logical console.
                     50    01   D0   0630    950          MOVL    #SS$_NORMAL,R0              ; Signal success.
                           04   04   0633    951 10$:     RET
                                0634    952
                                0634    953          .END    ERF$START
```

```
$$.TAB                          = 0000067C R      02
$$.TABEND                       = 000006DC R      02
$$.TMP                          = 02000000
$$.TMP1                         = 00000001
$$.TMP2                         = 000000CF
$$.TMPX                         = 00000000 R      03
$$.TMPX1                        = 00000018
$$T1                            = 00000000
$$T2                            = 00000003
..AFLG                          = 00000000
..FLG                           = 00000000
..MOD                           = 00000001
..N                             = 00000001
..TYP                           = 00000003
.LEN                            = 00000001
BYEMSG                            00000462 R      02
BYEMSG_DSC                        0000045A R      02
BYEMSG_END                        000004E2 R      02
BYEMSG_LEN                        0000045A R      02
CON$C_INVSNP1                   = 00000031
CON$C_INVSNP2                   = 00000032
CON$C_REQERL                    = 00000030
CON$SENDCONSCMD                   ******** X    04
DC$_DISK                        = 00000001
DC$_TAPE                        = 00000002
DEVFAO                            00000305 R      02
DIB$W_DEVBUFSIZ                 = 00000006
DVI$_DEVCLASS                   = 00000004
EMB$B_VALID                     = FFFFFFFF
EMB$C_HD_LENGTH                 = 00000010
EMB$C_HLT                       = 00000010
EMB$C_TS                        = 00000026
EMB$C_TS_LENGTH                 = 00000010
EMB$C_VD                        = 00000041
EMB$C_VM                        = 00000040
EMB$K_HD_LENGTH                 = 00000010
EMB$K_LENGTH                    = 00000004
EMB$K_NF                        = 00000023
EMB$K_TS                        = 00000026
EMB$L_HD_SID                    = 00000000
EMB$Q_HD_TIME                   = 00000006
EMB$W_HD_ENTRY                  = 00000004
EMB$W_HD_ERRSEQ                 = 0000000E
EMB$W_SIZE                      = FFFFFFFC
ERF$B_ERRCNT                      000005BD R      02
ERF$B_HD_DCLASS                   00000004
ERF$B_HD_DTYPE                    00000005
ERF$B_MAXERRCNT                   000005BE R      02
ERF$B_VM_NAMLNG                   0000001E
ERF$C_HD_LENGTH                   00000010
ERF$C_LOOP_CNT                  = 000000FF
ERF$C_TS_LENGTH                   00000010
ERF$ERRSNAP                       0000054A RG     04
ERF$GETBUF                        000004AB RG     04
ERF$INIT                          00000352 RG     04
ERF$K_CLK_TICK                  = FF676980
ERF$K_DLTA_STMP                 = 00000258
ERF$K_HD_LENGTH                   00000010
ERF$K_TS_LENGTH                   00000010
ERF$L_HD_SID                      00000000
ERF$L_VM_ERRCNT                   00000014
ERF$L_VM_OPRCNT                   00000018
ERF$L_VM_OWNUIC                   00000010
ERF$MOUNT                         00003B9 RG      04
ERF$M_HD_INVALD                 = 00000080
ERF$Q_DELTA                       00000008 R      04
ERF$Q_HD_TIME                     00000006
ERF$Q_WAIT                        00000010 R      04
ERF$SNAPSHOT_COPIED               000005E8 RG     04
ERF$SNAPSHOT_PRESENT              000005D4 RG     04
ERF$START                         0000002F RG     04
ERF$TIMSTMP                       0000038E RG     04
ERF$T_VM_LABEL                    00000032
ERF$T_VM_NAMTXT                   0000001F
ERF$W_HD_ENTRY                    00000004
ERF$W_HD_ERRSEQ                   0000000E
ERF$W_MBXCHN                      000002FF R      02
ERF$W_MBXSIZ                      00000301 R      02
ERF$W_MBXUNT                      00000303 R      02
ERF$W_VM_NUMSET                   0000C030
ERF$W_VM_UNIT                     0000001C
ERF$W_VM_VOLNUM                   0000002E
ERL$ALLOCEMB                      ******** X    04
ERL$AL_BUFADDR                    ******** X    04
ERL$B_BUSY                      = 00000000
ERL$B_FLAGS                     = 00000003
ERL$B_MSGCNT                    = 00000001
ERL$C_LENGTH                    = 0000000C
ERL$GB_BUFPTR                     ******** X    04
ERL$GL_ERLPID                     ******** X    04
ERL$L_END                       = 00000008
ERL$L_NEXT                      = 00000004
ERL$RELEASEMB                     ******** X    04
ERL$V_LOCK                      = 00000000
ERM$C_FORMAT                    = 00000002
ERRSNAP_COM                       000007DB R      02
ERRSNAP_CONCMD                    00000841 R      02
ERRSNAP_DATA                      00000842 R      02
ERRSNAP_FAB                       00000600 R      02
ERRSNAP_FLAGS                     00000839 R      02
ERRSNAP_LOG1                      000007FB R      02
ERRSNAP_LOG2                      0000081A R      02
ERRSNAP_NAM                       0000067C R      02
ERRSNAP_RSA                       000006DC R      02
ERRSNAP_STATUS                    0000083D R      02
ERRSNAP_XAB                       00000650 R      02
EXE$GB_CPUTYPE                    ******** X    04
EXE$GL_MSGFLAGS                   ******** X    04
EXE$GQ_ERLMBX                     ******** X    04
EXE$IPID_TO_EPID                  ******** X    04
EXE$V_DISMOUMSG                   ******** X    04
EXE$V_MOUNTMSG                    ******** X    04
FAB$B_FNS                       = 00000034
FAB$B_ORG                       = 0000001D
```

```
FAB$B_RFM                       =  0000001F
FAB$B_SHR                       =  00000017
FAB$C_BID                       =  00000003
FAB$C_BLN                       =  00000050
FAB$C_SEQ                       =  00000000
FAB$C_VAR                       =  00000002
FAB$L_ALQ                       =  00000010
FAB$L_FNA                       =  0000002C
FAB$L_FOP                       =  00000004
FAB$V_CHAN_MODE                 =  00000002
FAB$V_CIF                       =  00000019
FAB$V_FILE_MODE                 =  00000004
FAB$V_GET                       =  00000001
FAB$V_LNM_MODE                  =  00000000
FAB$V_PUT                       =  00000000
FAB$V_UPD                       =  00000003
FAB$V_UPI                       =  00000006
FAB$W_GBC                       =  00000048
FAB$W_IFI                       =  00000002
FAB$W_MRS                       =  00000036
FILCRE                            00000000 R      04
INBUF                             00000000 R      02
IO$M_NOW                          ********     X  04
IO$_WRITEVBLK                     ********     X  04
LASTENTRY                         000002FA R      02
LIB$SPAWN                         ********     X  04
MBX                               00000239 R      04
MOUNT_BUF                         00000528 R      02
MOUNT_DMT                         000005B1 R      02
MOUNT_DSC                         00000520 R      02
MOUNT_END                         00000520 R      02
MOUNT_FAO                         000004E2 R      02
MOUNT_MNT                         000005A8 R      02
MOUNT_MSG                         000004EA R      02
MSGOUT                            00000162 R      04
NAM$B_ESS                       =  0000000A
NAM$B_NOP                       =  00000008
NAM$B_RSL                       =  00000003
NAM$B_RSS                       =  00000002
NAM$C_BID                       =  00000002
NAM$C_BLN                       =  00000060
NAM$C_MAXRSS                    =  000000FF
NAM$L_ESA                       =  0000000C
NAM$L_RSA                       =  00000004
NAM$W_FID                       =  00000024
NAMEBLOCK                         00000294 R      02
NXTMSG                            00000139 R      04
OPC$M_NM_CENTRL                 =  00000001
OPC$M_NM_DEVICE                 =  00000010
OPC$M_NM_DISKS                  =  00000008
OPC$M_NM_TAPES                  =  00000004
OPC$_RQ_RQST                    =  00000003
OPRMSG                            00000325 R      02
OPRMSG_DSC                        00000315 R      02
OPRMSG_END                        00000356 R      02
OPRMSG_LEN                        00000315 R      02
OUTFAB                            00000200 R      02
OUTFID                            000002F4 R      02
OUTNAM                            00000018 R      04
OUTNAMSZ                        =  00000017
OUTRAB                            00000250 R      02
PCB$L_PID                       =  00000060
PR$_IPL                         =  00000012
PR$_SID                         =  0000003E
PR$_SID_TYP790                  =  00000004
PRCBUF                            0000004C R      04
PRCNXT                            00000070 R      04
PRCNXT1                           00000072 R      04
ROMSG                             00000356 R      02
ROMSG_DSC                         0000031D R      02
ROMSG_END                         00000456 R      02
ROMSG_LEN                         00000456 R      02
RAB$B_RAC                       =  0000001E
RAB$C_BID                       =  00000001
RAB$C_BLN                       =  00000044
RAB$C_RFA                       =  00000002
RAB$C_SEQ                       =  00000000
RAB$L_CTX                       =  00000018
RAB$L_RBF                       =  00000028
RAB$L_ROP                       =  00000004
RAB$V_EOF                       =  00000008
RAB$V_WBH                       =  0000000A
RAB$W_ISI                       =  00000002
RAB$W_RSZ                       =  00000022
RETURN_STATUS                     000005D3 R      04
SCH$GL_CURPCB                     ********     X  04
SID                               000002FB R      02
SNDMSG                            000003DA R      04
SS$_NORMAL                      =  00000001
SYS$ASSIGN                        ********    GX  04
SYS$CLOSE                         ********    GX  04
SYS$CMKRNL                        ********    GX  04
SYS$CONNECT                       ********    GX  04
SYS$CREATE                        ********    GX  04
SYS$DASSGN                        ********    GX  04
SYS$DELPRC                        ********    GX  04
SYS$FAO                           ********     X  04
SYS$FIND                          ********    GX  04
SYS$GETCHN                        ********    GX  04
SYS$GETDVI                        ********    GX  04
SYS$GETMSG                        ********    GX  04
SYS$HIBER                         ********    GX  04
SYS$OPEN                          ********    GX  04
SYS$PUT                           ********    GX  04
SYS$QIO                           ********    GX  04
SYS$SETIMR                        ********    GX  04
SYS$SNDOPR                        ********    GX  04
SYS$UPDATE                        ********    GX  04
SYS$WAITFR                        ********    GX  04
WRITE_FAILURE                     000001BB R      04
XAB$C_DAT                       =  00000012
XAB$C_DATLEN                    =  0000002C
XAB$L_NXT                       =  00000004
XAB$Q_CDT                       =  00000014
```

XAB$Q_EDT = 0000001C

Psect synopsis

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
ABS	0000003E (62.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
DATA	00000846 (2118.)	02 (2.)	NOPIC USR CON REL LCL NOSHR NOEXE RD WRT NOVEC PAGE
$RMSNAM	00000018 (24.)	03 (3.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
CODE	00000634 (1588.)	04 (4.)	NOPIC USR CON REL LCL NOSHR EXE RD NOWRT NOVEC BYTE

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	29	00:00:00.08	00:00:00.36
Command processing	156	00:00:00.86	00:00:03.07
Pass 1	559	00:00:23.26	00:00:50.17
Symbol table sort	0	00:00:02.71	00:00:04.76
Pass 2	191	00:00:04.47	00:00:09.16
Symbol table output	28	00:00:00.19	00:00:00.33
Psect synopsis output	2	00:00:00.03	00:00:00.03
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	967	00:00:31.61	00:01:07.88

The working set limit was 1950 pages.
119098 bytes (233 pages) of virtual memory were used to buffer the intermediate code.
There were 100 pages of symbol table space allocated to hold 1834 non-local and 42 local symbols.
953 source lines were read in Pass 1, producing 45 object records in Pass 2.
87 pages of virtual memory were used to define 71 macros.

Macro library statistics

Macro library name	Macros defined
_255DUA28:[ERRFMT.OBJ]ERRFMT.MLB;1	3
_255DUA28:[SYS.OBJ]LIB.MLB;1	9
_255DUA28:[SYSLIB]STARLET.MLB;2	56
TOTALS (all libraries)	68

2513 GETS were required to define 68 macros.

There were no errors, warnings or information messages.

MACRO/LIS=LIS$:ERRFMT/OBJ=OBJ$:ERRFMT MSRC$:ERRFMT/UPDATE=(ENH$:ERRFMT)+EXECML$/LIB+LIB$:ERRFMT/LIB

EVLDEF
MDL

LIBTAIL
B32

XXXQUE
LIS

UTILDEF
REQ

ERRDEF
MDL

EVCDEF
MDL

CONSOLE
LIS

USEROPEN
LIS

VAXPSL
LIS

ERRFMT

VAX780REG
LIS

ERRFMT
MAP

EVLIBRARY
B32

VAX750REG
LIS

VECMAPREG
LIS

ERRFMT
LIS

EVL

LIBHEAD
B32

EVL
MAP

WQDEF
MDL